The Processing Graph Method Tool (PGMT)

Richard S. Stevens

15 July 1997

U.S. Naval Research Laboratory

Washington, DC

stevens@ait.nrl.navy.mil

http://www.ait.nrl.navy.mil/pgmt/pgm2.html

Project Leader: David Kaplan

Outline

- Introduction
- Processing Graph Method (PGM)
- PGM Tool (PGMT)
- Current Status
- Conclusions

Introduction

- Demands for throughput outpacing performance
- Turning to distributed systems
  - Software for distributed system must be tailored
    - Requires team-work: application engineer with expert programmer
  - Port to new architecture requires complete rewrite
  - Application software very expensive
  - Limited access to distributed systems
- Needs
  - Architecture independent language for concurrent processing
    - Expose concurrency
    - Intuitive, graphic
    - Easy development & maintenance
  - Tool set to build low cost compilers
    - Target the language to new distributed architectures
    - Automate the tailoring for target architecture
    - Easy, inexpensive porting of application software

Processing Graph Method (PGM)

Reconfigurable data flow

- Directed bipartite graph (like Petri net)
- Places for data storage
- Transitions for processing
- Directed edges to indicate flow of data
- Family for multiplicity
- Included graphs for
  - Hierarchical structure
  - Modularity
  - Reuse
- Command program for reconfiguration

Place

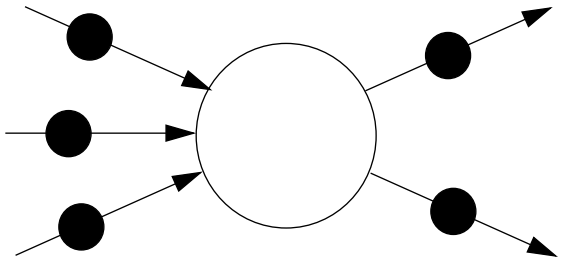
- **Mode specifies data type to be stored**
  - **Data unit called a *token***
- **Queue**
  - **FIFO storage with capacity (max token count)**
  - **Unique producer**
  - **Unique consumer**
- **Graph variable**
  - **Single token storage**
  - **Multiple producers**
  - **Multiple consumers**
  - **New data overwrites old**
  - **Consumer gets most recent value**

Family

- List of members of a common type
- Each member is (recursively)
  - A family
  - A base type (int, float, queue, transition, ...)
- Members have
  - Same base type
  - Same depth of recursion
  - Possibly different family sizes

Transition

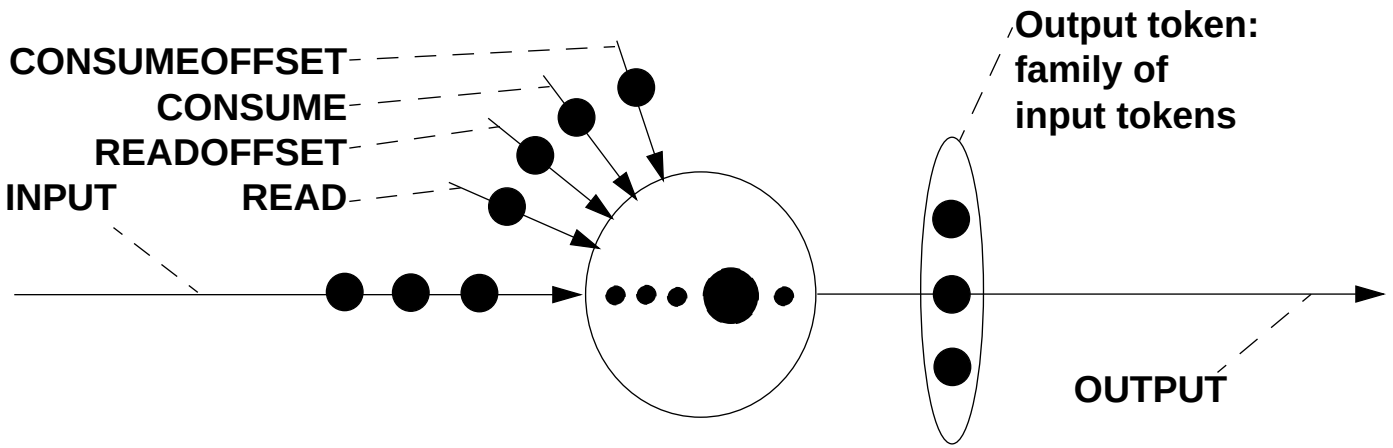
- **Execution requires**
  - **Sufficient input data**
  - **Sufficient capacity for output data**
- **No internal memory between executions**
- **Ordinary transition**
  - **Read & consume one token at each input in each execution**
    - May input a large (family-size) token
  - **Produce one token at each output**
    - May output a large (family-size) token
  - **User-written transition statement**
    - Specifies processing
    - May call primitives



Special Transitions

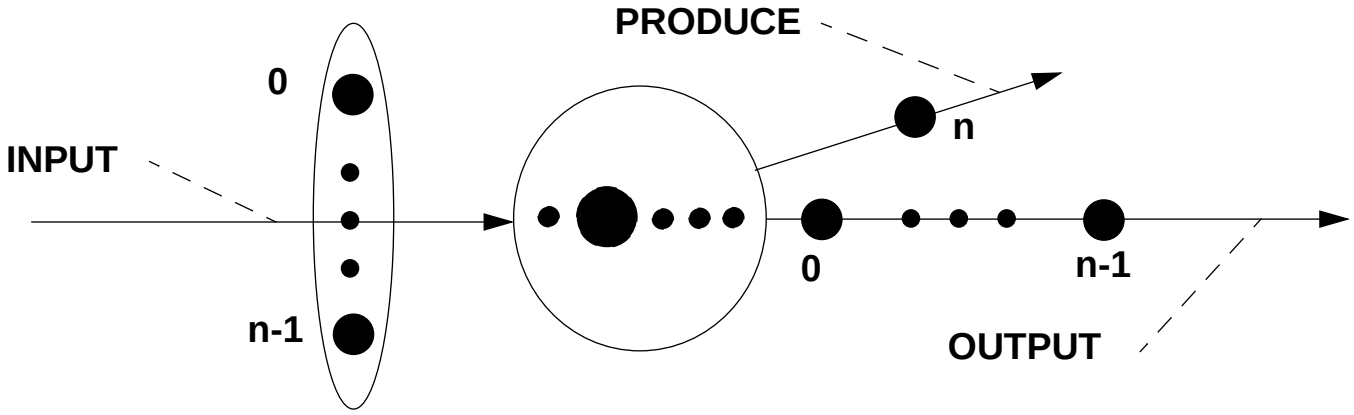
- May consume any number of input tokens
- May produce any number of output tokens
- Reformat data - no processing
- Specified by PGM
  - Pack
  - Unpack
  - Uncontrolled Merge
  - Others, to be added as needed

Pack Transition



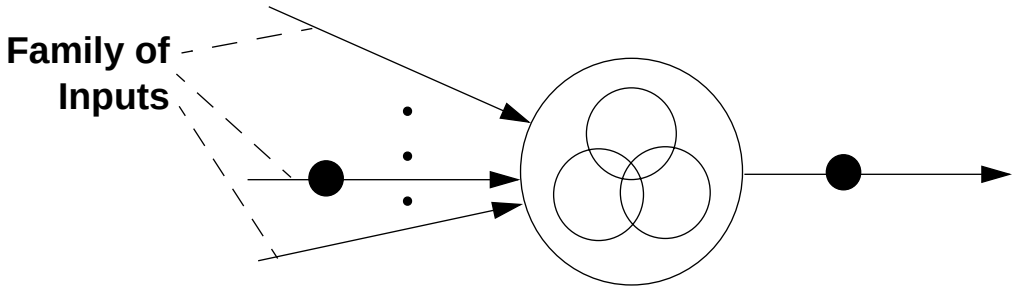
Pack transition: Assemble a specified number of input tokens into a family and output that as a single token.

Unpack Transition



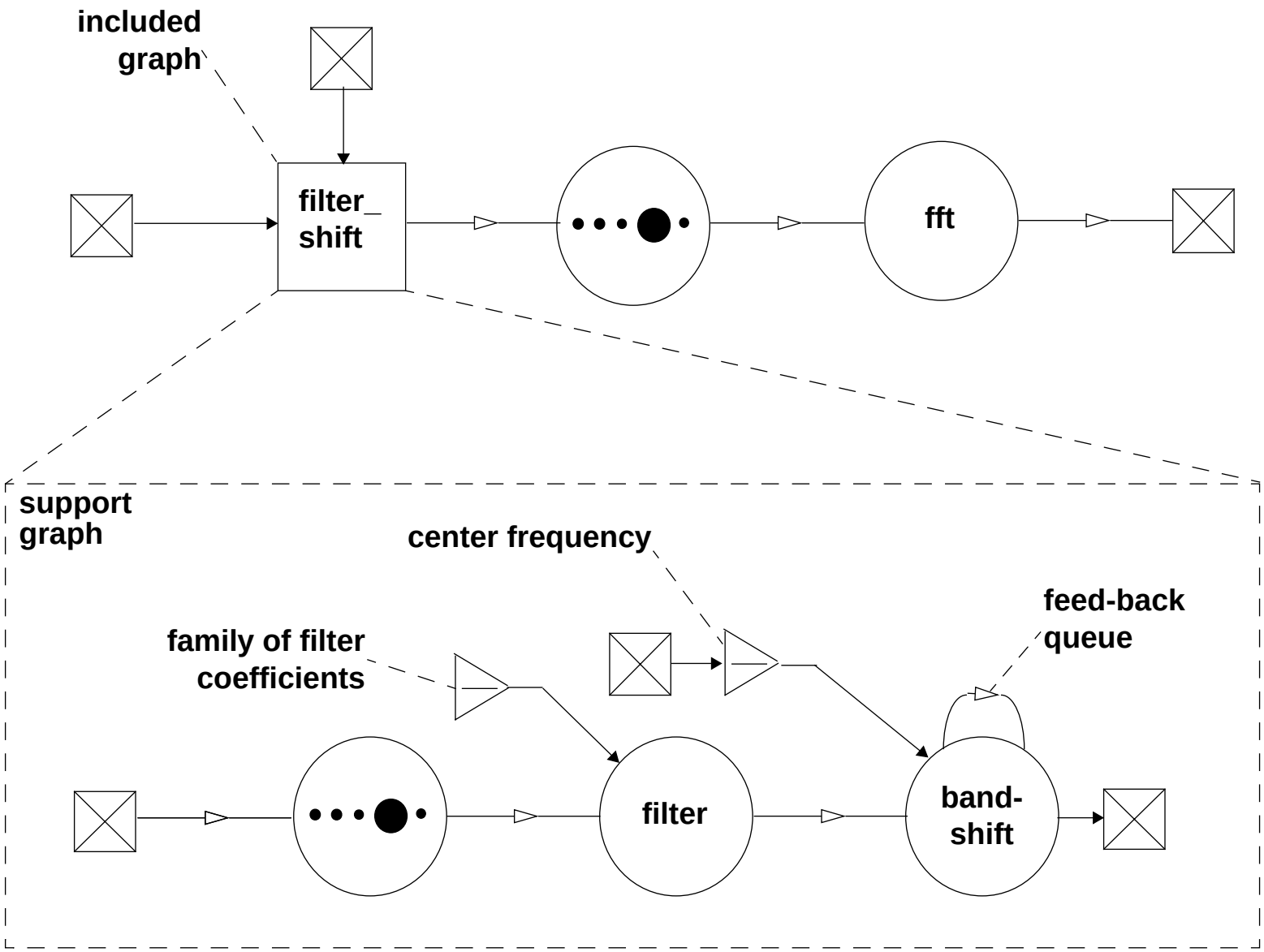
Unpack transition: Disassemble an input token that is a family and output the individual members as tokens.

Uncontrolled Merge



Uncontrolled Merge: Input one token (the first available) from one of a family of inputs and pass it on.

Example of a Processing Graph



Configuration & Reconfiguration

- **Command program**
  - **Adapt structure of processing graph in response to changing environment**
- **Procedure library provides capability to**
  - **Create a processing graph and enable transitions to execute**
  - **Input data into a graph**
  - **Change values of parameters by input to a graph variable**
  - **Read output data from a graph**
  - **Suspend processing**
  - **Change graph structure by disconnecting and reconnecting places and transitions**

PGMT - The Project

- **Demonstrate implementation of PGM on distributed architectures**
- **Provide a tool set to**
  - **Capture (by GUI) the target architecture**
  - **Analyze the target architecture**
  - **Capture application processing graphs**
  - **Analyze processing graphs**
  - **Partition processing graph into clusters**
  - **Assign the clusters to processors in the target architecture**

Inputs to Architecture Analysis

- Number of each kind of processor
- Primitives that will run on each kind of processor
- Execution time of each primitive on each kind of processor
- Communication paths and times between processors

Technical Problems

- **Assignment and scheduling**
  - **Partition into clusters**
  - **Static assignment and scheduling within each cluster**
  - **Run-time assignment and scheduling of the clusters**
  - **Reconfiguration implies subsequent analysis, partition, assignment, and scheduling**
  - **Architecture reconfiguration implies subsequent analysis, partition, assignment, and scheduling**
- **Achieving highest throughput is NP hard**
  - **Heuristics to find suboptimal solution**

PGMT - Current Status of Project

- **Achievements to date**
  - **Graph specification format defined (ASCII text)**
  - **GUI capture of processing graph**
  - **Implement on single processor**
- **Current effort**
  - **Target fixed homogeneous network architecture**
- **Plans**
  - **Target fixed heterogeneous network**
  - **Target reconfigurable heterogeneous network**

Conclusions

- **PGM**
  - **Architecture independent**
  - **Reconfigurable data flow**
  - **Application engineer can work alone**
    - **Expert programmer no longer needed**
- **PGMT**
  - **To demonstrate a tool set to implement PGM**
    - **On a wide variety of distributed architectures**
    - **At relatively low cost**
  - **All software in public domain**
  - **Expect commercial efforts to continue work**
  - **If successful, all will benefit**
    - **Broader access to distributed systems**